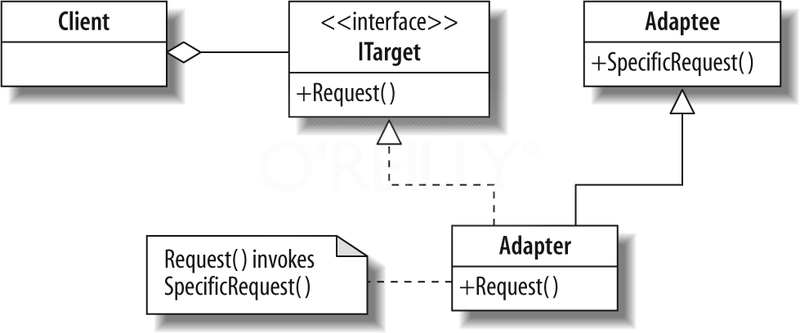
# Adapter design pattern:

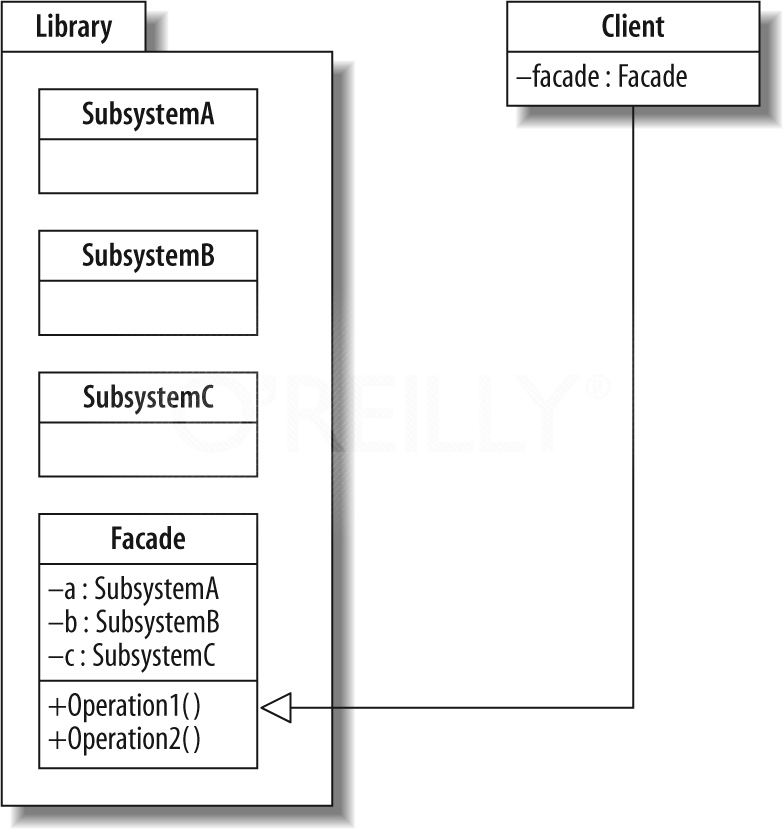


Adapter design pattern-а се използва главно с цел да приспособи един клас, модул или приложение да работи с друг клас/модул/приложение без това да е било заложено първоначално в дизайна на едното от двете.

Много често adapter design pattern-а се използва за по-лесно справяне с legacy code – т.е. стар код, който трябва да се използва в приложението и няма възможност за пренаписването му. Видовете адаптор могат да бъдат:

* Class
* Object
* Two-way
* Pluggable

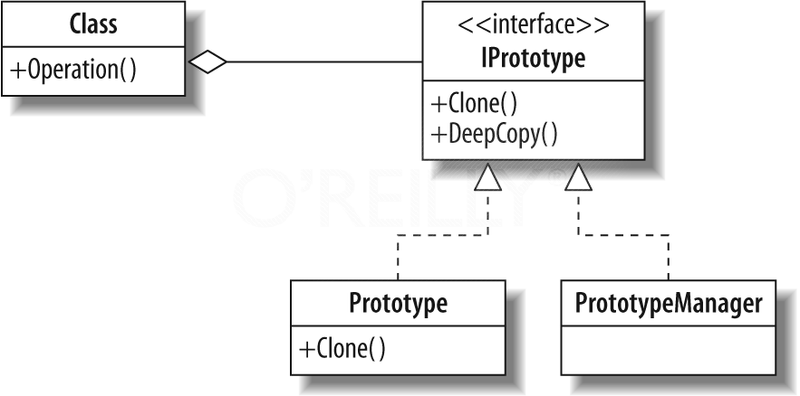
# Facade design pattern.



Facade design pattern се използва главно, за да скрива вътрешната бизнес логика на сложна система, така че да предоставя лесна работа на външните приложения и достъп, без да се налага те да са наясно как системата работи вътрешно.

Пример: Приложение , което работи с много по вид бази данни – oracle, SQL, MySQL, Mongo DB и тн. Тъй като не искаме да имплементираме логика за работя с всяка една база данни, по-уместно би било да извадим тази логика в Facade класа, който единствен да бъде достъпван отвън и само той да се грижи за логиката при работа с базите данни.

# Prototype pattern.



Prototype design pattern-a се използва най-често при нужда от създаване на много обекти от един и същи тип с еднаква показатели. Големите му предимства са, че съзадва не нова инстанция на класа, ами клонира даден обект като копира всичките му пропъртита.

Удобно е също и, че създаденият обект е напълно автономен и може да бъде променян, без това да повлияе на прототипа му.

Source for the info and the code:

<http://msdn.microsoft.com/en-us/library/orm-9780596527730-01.aspx>